**Experiment : 1**

**DESIGN AND SIMULATION OF ADDERS USING VERILOG HDL**

**HALF ADDER**

**Half Adder program:**

module half\_adder(a,b,s,c);

input a,b;

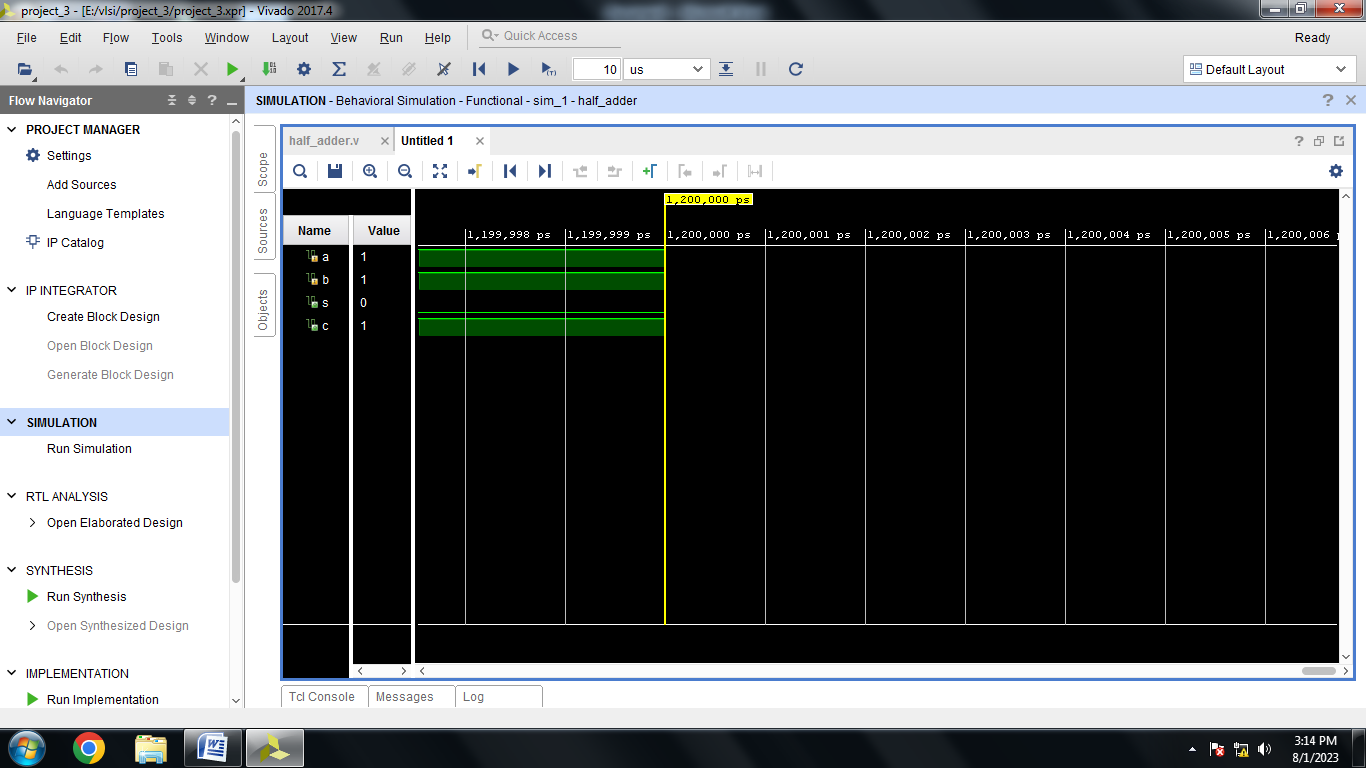
output s,c;

xor x1(s,a,b);

and x2(c,a,b);

endmodule

**Half Adder Output:**



**FULL ADDER**

**Full Adder Program:**

module full\_adder(a,b,c,sum,carry);

input a,b,c;

output sum,carry;

wire w1,w2,w3;

xor x1(sum,a,b,c);

and x2(w1,a,b);

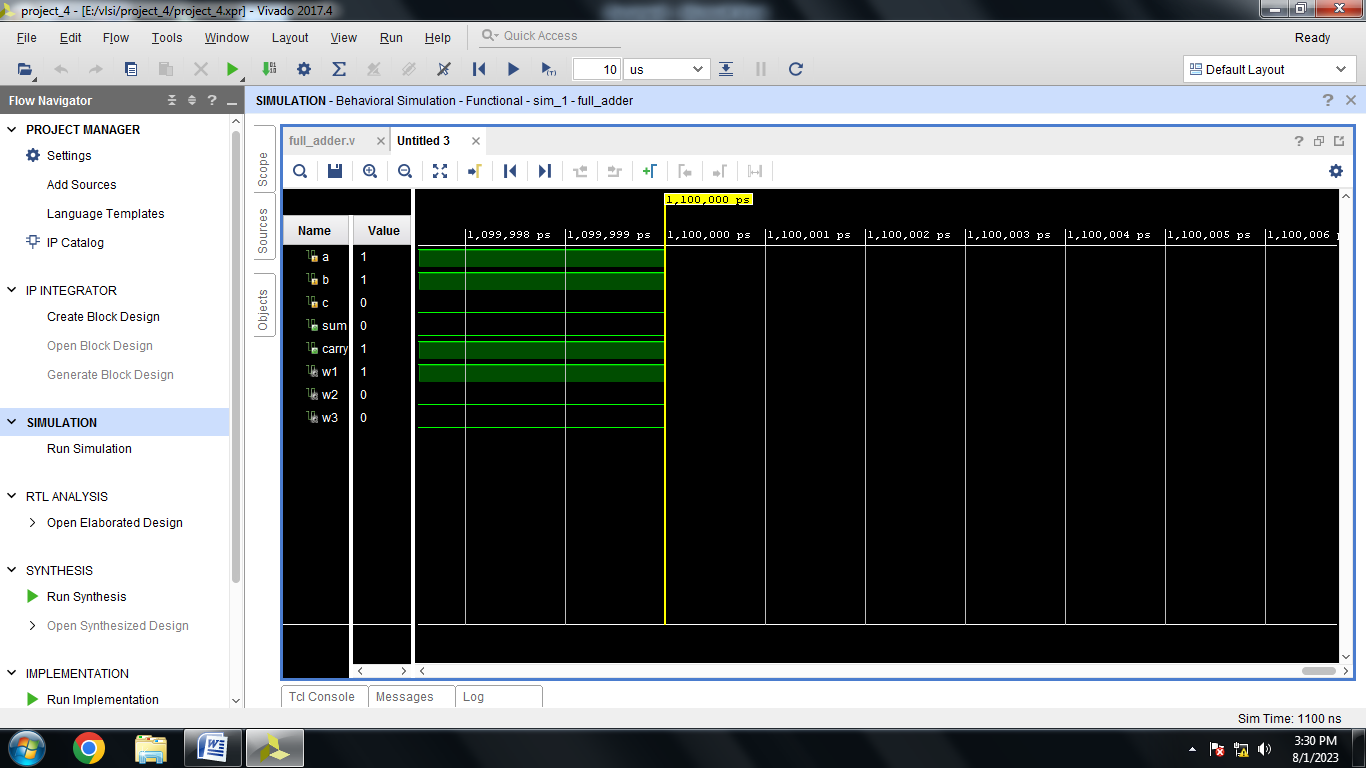
and x3(w2,b,c);

and x4(w3,c,a);

or x5(carry,w1,w2,w3);

endmodule

**Full Adder Output:**



**RIPPLE CARRY ADDER**

**Ripple Carry Adder Program:**

module full\_adder(a,b,c,sum,carry);

input a,b,c;

output sum,carry;

wire w1,w2,w3;

xor x1(sum,a,b,c);

and x2(w1,a,b);

and x3(w2,b,c);

and x4(w3,c,a);

or x5(carry,w1,w2,w3);

endmodule

module rca(

input [3:0] a,

input [3:0] b,

input c,

output [3:0] sum,

output carry

);

wire w1,w2,w3;

full\_adder f0(a[0],b[0],c,sum[0],w1);

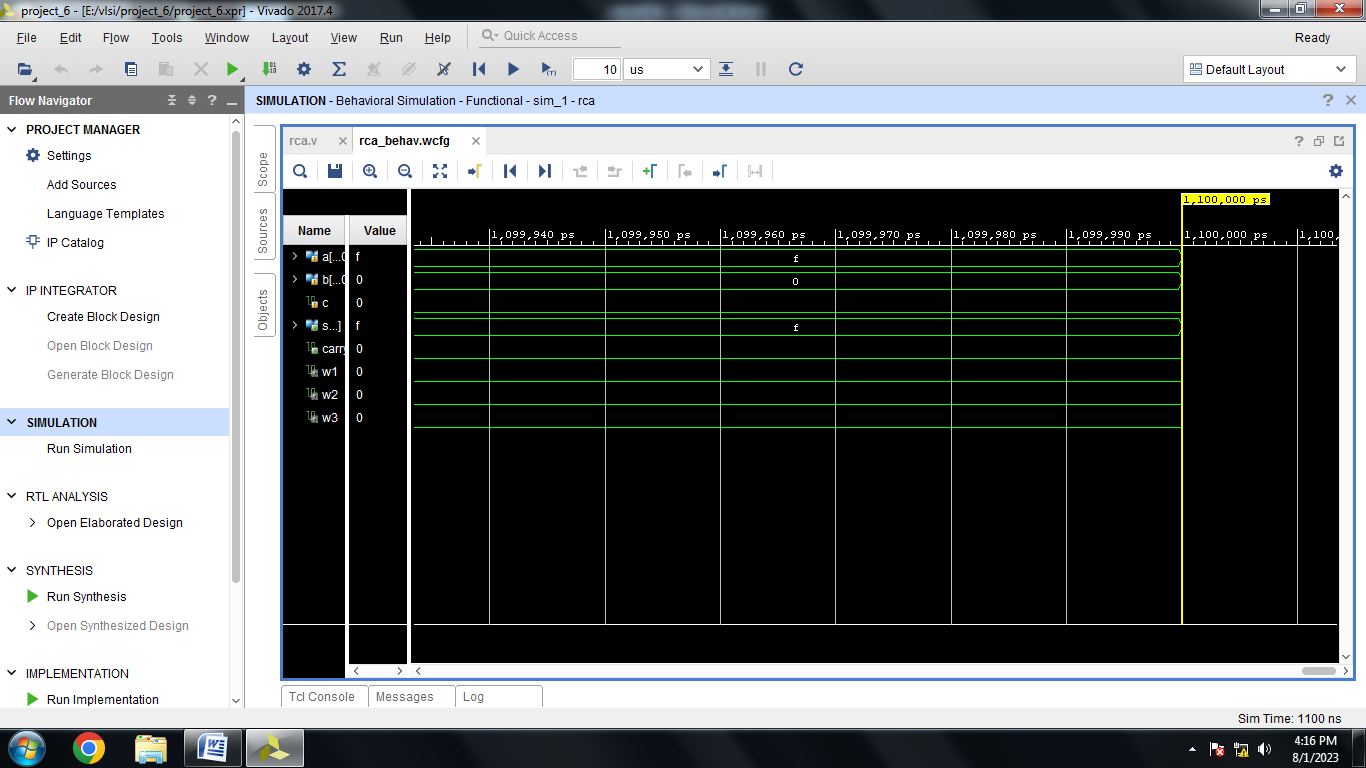
full\_adder f1(a[1],b[1],w1,sum[1],w2);

full\_adder f2(a[2],b[2],w2,sum[2],w3);

full\_adder f3(a[3],b[3],w3,sum[3],carry);

endmodule

**Ripple Carry Adder Output:**



**Experiment 2**

**DESIGN AND SIMULATION OF FLIP-FLOPS USING VERILOG HDL**

**SR FLIP FLOP**

**SR Flip Flop Program:**

module srff(

input s,

input r,

input clk,

input reset,

output reg q,qb

);

always@(posedge clk)

begin

if((s==0)&&(r==0))

q=q;

else if((s==0)&&(r==1))

q=0;

else if((s==1)&&(r==0))

q=1;

else

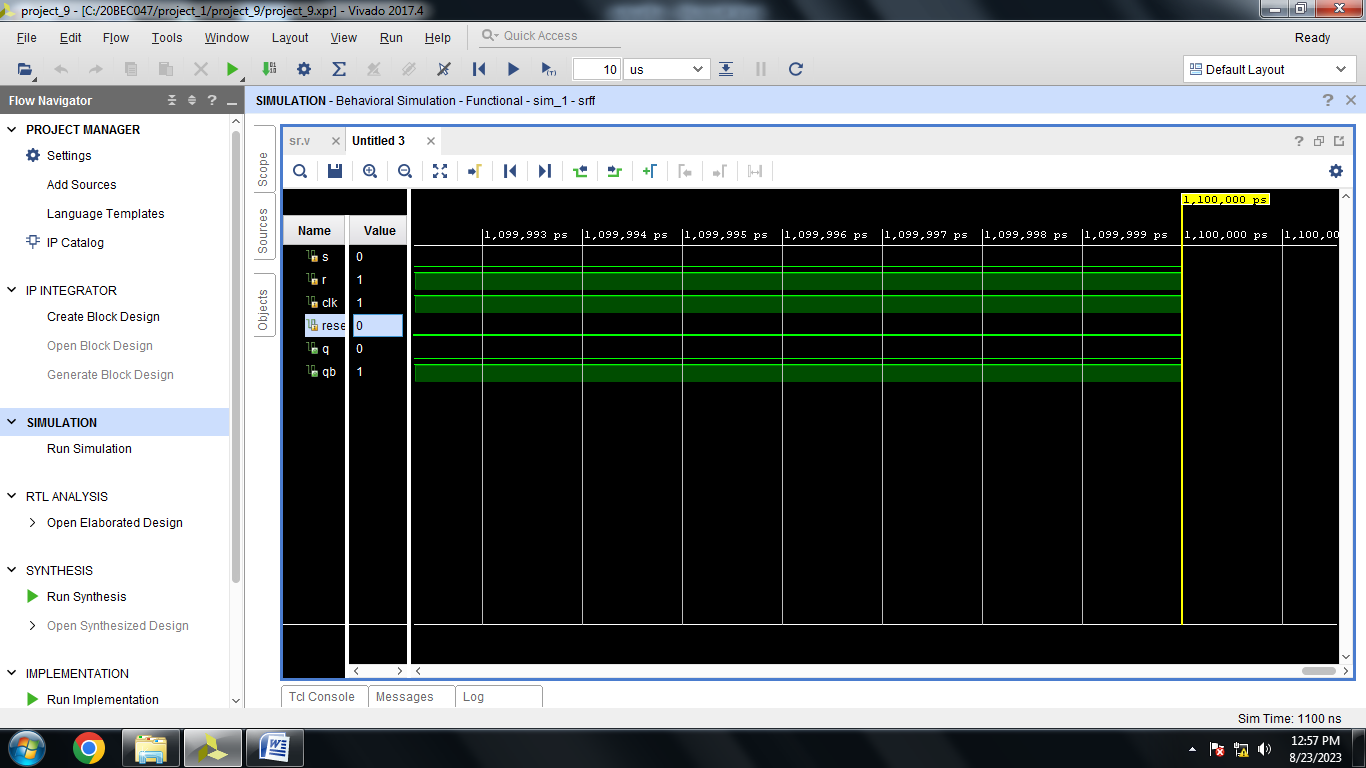
q=~q;

qb<=~q;

end

endmodule

**SR Flip Flop Output:**

****

**JK Flipflop**

**JK Flipflop Program:**

module flipflop(j,k,clk,reset,q);

input j,k,clk,reset;

output q;

reg q;

always@(posedge clk or posedge reset)

begin

if(reset)

q<=1'b0;

else

case({j,k})

2'b00:q<=q;

2'b01:q<=1'b0;

2'b10:q<=1'b1;

2'b11:q<=~q;

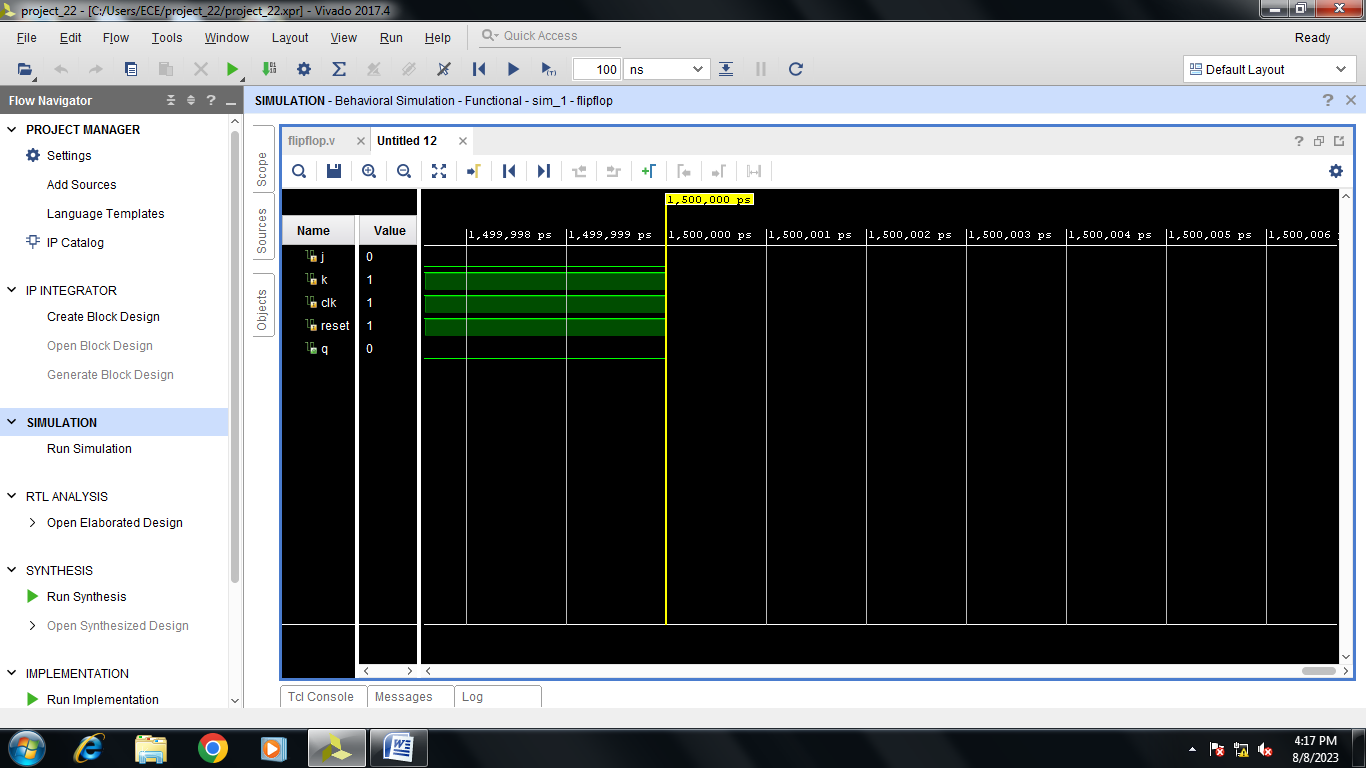
default:q<=q;

endcase

end

endmodule

**JK Flipflop Output:**

****

**D Flip Flop**

**D Flip Flop Program:**

module Dflipflop(d,clk,rst,q);

input d,clk,rst;

output reg q;

always @ (posedge clk)

begin

if(rst)

q<=1'b0;

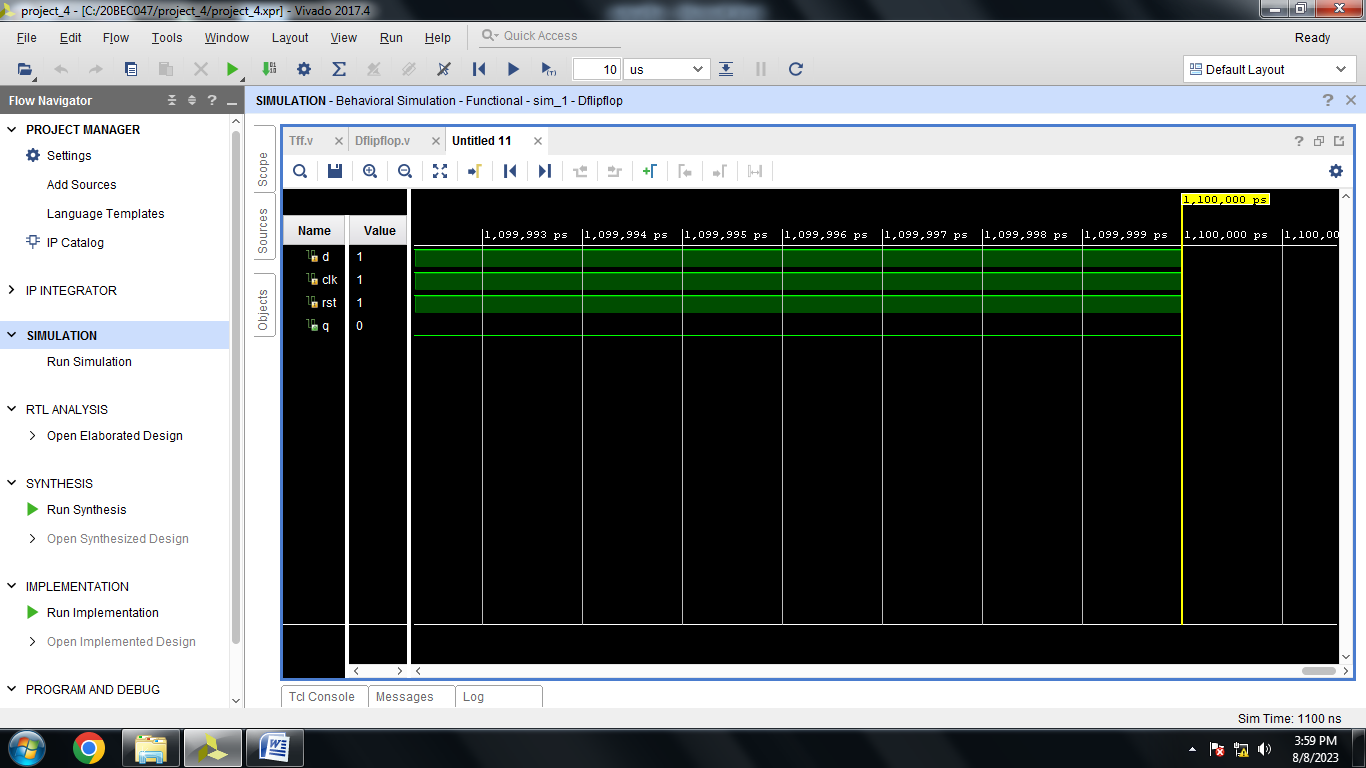
else

q<=d;

end

endmodule

**D Flip Flop OUTPUT:**

****

**T FLIP FLOP**

**T Flip Flop Program:**

module Tff(t,clk,rst,q);

input t,clk,rst;

output q;

reg q;

always @(posedge rst)

begin

if(rst)

q<=1'b0;

else

if(t==1'b1)

q<=~q;

else

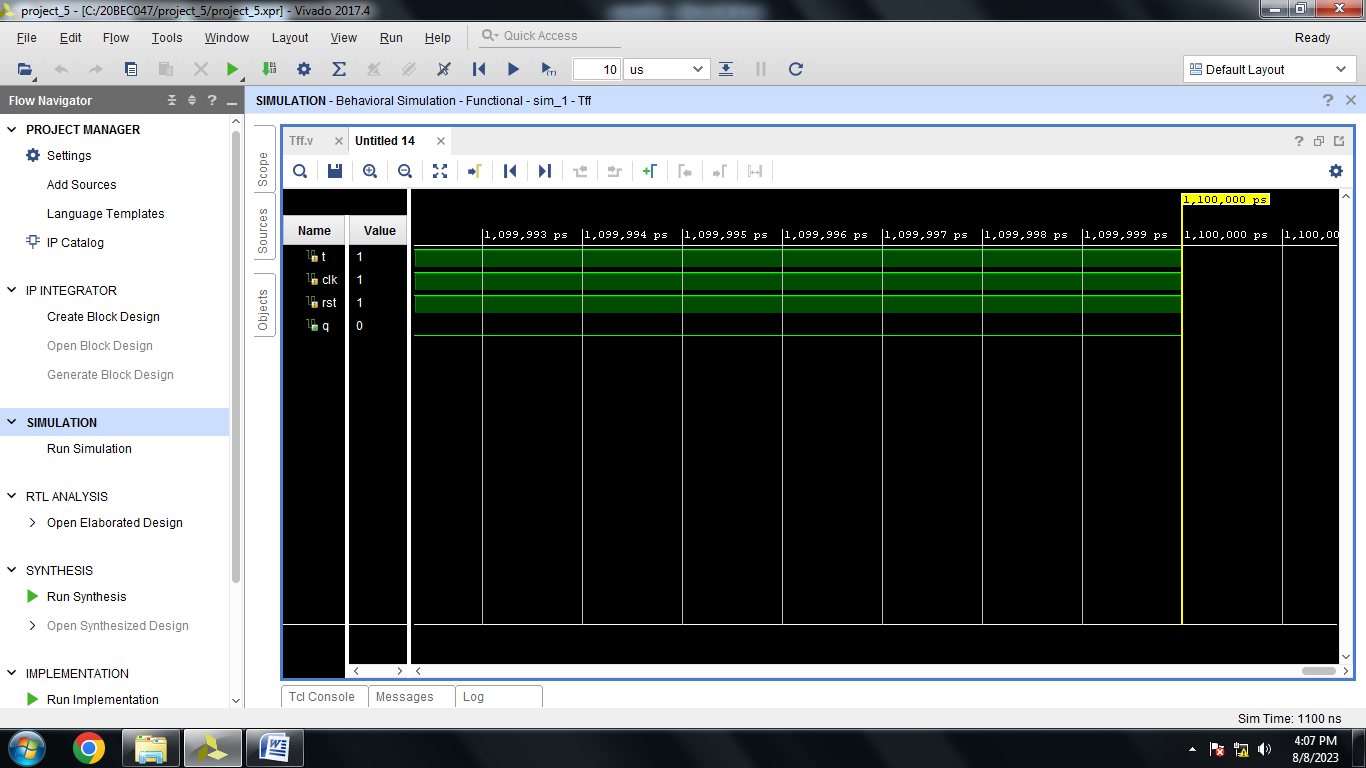
if(t==1'b0)

q<=q;

end

endmodule

**T Flip Flop Output:**

****

**Experiment 3**

**Design And Implementation Of 4 Bit Asynchronous Counterusing FPGA**

**4 Bit Asynchronous Counter Program:**

module async(

input j,

input k,

input clk,

input reset,

output wire [3:0]q,qb

);

jkff JK1(j,k,clk,reset,q[0],qb[0]);

jkff JK2(j,k,q[0],reset,q[1],qb[1]);

jkff JK3(j,k,q[1],reset,q[2],qb[2]);

jkff JK4(j,k,q[2],reset,q[3],qb[3]);

endmodule

module jkff(input j,

input k,

input clk,

input reset,

output reg q,qb);

always@(negedge clk)

begin

case({reset,j,k})

3'b100 :q=q;

3'b101 :q=0;

3'b110 :q=1;

3'b111 :q=~q;

default :q=0;

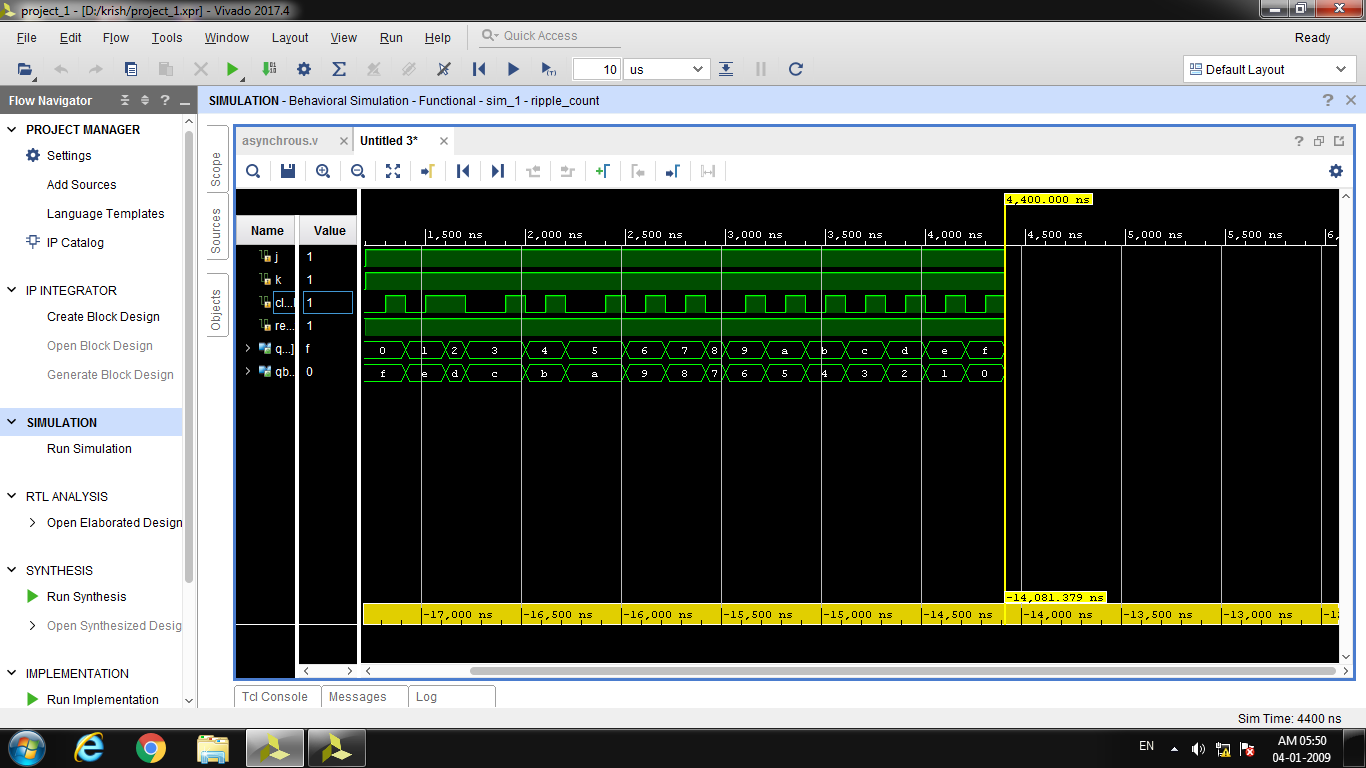
endcase

qb<=~q;

end

endmodule

**4 Bit Asynchronous Counter Output:**



**Experiment 4**

**Design And Implementation Of 4 Bit Synchronous Counterusing FPGA**

**4 Bit Synchronous Counter Program:**

module sync\_count(

input t,

input clk,

input reset,

output [3:0]q,qb

);

wire x1,x2;

tff T0(t,clk,reset,q[0],qb[0]);

tff T1(q[0],clk,reset,q[1],qb[1]);

and A1(x1,q[0],q[1]);

tff T2(x1,clk,reset,q[2],qb[2]);

and A2(x2,q[2],x1);

tff T3(x2,clk,reset,q[3],qb[3]);

endmodule

module tff(input t,

input clk,

input reset,

output reg q,qb);

always@(posedge clk)

begin

case({reset,t})

2'b00 :q=q;

2'b01 :q=~q;

default :q=0;

endcase

qb<=~q;

end

endmodule

**4 Bit Synchronous Counter Output:**

